http://www-lipn.univ-paris13.fr/~petrucci/PAPERS
*Petri Net Newsletter 68, April 2005*

# Cover Picture Story:
## Experiments with Modular State Spaces

Laure Petrucci

LIPN, CNRS UMR 7030, Université Paris XIII
99, avenue Jean-Baptiste Clément
F-93430 Villetaneuse, FRANCE
petrucci@lipn.univ-paris13.fr

**Abstract.** The main analysis techniques for critical systems use state space exploration. However, one is often quickly limited by the so-called *state space explosion* problem, even if the systems models are relatively small. Several methods have been proposed to tackle this problem in order to get a manageable state space. One of these takes advantage of the modular structure of the model in order to build a modular state space, which is basically a set of local state spaces plus a synchronisation graph indicating the global behaviour. In this paper, we present a few case studies and compare the state space sizes with those of the full state space. The results are discussed, pointing out the criteria the model should satisfy for the technique to be efficient.

## 1 Introduction

The use of high-level Petri net formalisms has made it possible to create Petri net models of large systems. Even though the use of such models allows the modeller to create compact representations of data and action, the size of models has been increasing. A large model can make it difficult to handle the complexity of the modelling as well as the analysis of the total system. It is well-known that the use of a modular approach to modelling has a lot of advantages: it allows the modeller to consider different parts of the model independently of one another. A modular approach to analysis is also attractive: it often dramatically decreases the complexity of the analysis task.

The main analysis technique consists in building the occurrence graph and then check properties on this graph. However, one often has to cope with the so-called *state space explosion* problem. Several techniques have been designed in order to reduce the state space so that it can become manageable, e.g. partial order reductions [Hol91], sweep-line [CKM01], occurrence graphs with equivalences/symmetries [Jen94], ... The Modular State Space technique [CP00,LP04] takes advantage of the modular organisation of the model. This paper aims at evaluating the pros and cons of Modular State Spaces.

This paper is structured as follows. Section 2 recalls the basic notions of Modular State Spaces. In order to make experiments, several case studies are presented in section 3. Their Modular State Spaces size are compared with the occurrence graph size in section 4. These results are then discussed.

## 2 Modular State Spaces

In this section, we recall the basic definitions of modular Petri nets and modular state spaces. The reader already familiar with these or primarily interested in the experimental results can skip this section.

### 2.1 Modular Petri Nets

Modular Petri nets considered here consist only of modules synchronised through shared transitions.

**Definition 1 ([LP04], definition 3).** *A* modular Petri net *is a pair $MN = (S, TF)$, satisfying:*

1. *$S$ is a finite set of* modules *such that:*
   - *Each module, $s \in S$, is a Petri net: $s = (P_s, T_s, W_s, M_{0_s})$.*
   - *The sets of nodes corresponding to different modules are pair-wise disjoint: $\forall s_1, s_2 \in S : [s_1 \neq s_2 \Rightarrow (P_{s_1} \cup T_{s_1}) \cap (P_{s_2} \cup T_{s_2}) = \emptyset]$.*
   - *$P = \bigcup_{s \in S} P_s$ and $T = \bigcup_{s \in S} T_s$ are the sets of all places and all transitions of all modules.*
2. *$TF \subseteq 2^T \setminus \{\emptyset\}$ is a finite set of non-empty* transition fusion sets.

*Explanation*
(1) A modular Petri net contains a finite set of modules, each of them being a Petri net. These modules must have disjoint sets of nodes.
(2) Each transition fusion set is a set of transitions to be fused (synchronised) together.

In the following, $TF$ also denotes $\cup_{tf \in TF} tf$.

**Definition 2 ([LP04], definition 4).** *A* transition group *$tg \subseteq T$ consists of either a single non-fused transition $t \in T \setminus TF$ or all members of a transition fusion set $tf \in TF$. The set of transition groups is denoted by $TG$.*

A transition group corresponds to a synchronised action. The arc weight function $W$ is extended to transition groups, i.e. $\forall p \in P, \forall tg \in TG$:

$$W(p, tg) = \sum_{t \in tg} W(p, t), \; W(tg, p) = \sum_{t \in tg} W(t, p).$$

Markings of modular Petri nets are defined as markings of Petri nets, over the set $P$ of all places of all modules. The restriction of a marking $M$ to a module $s$ is denoted by $M_s$.

**Definition 3 ([LP04], definition 5).** *A transition group $tg$ is* enabled *in a marking $M$, denoted by $M[tg\rangle$, iff $\forall p \in P : W(p, tg) \leq M(p)$.*
*When a transition group $tg$ is enabled in a marking $M_1$ it may* occur*, changing the marking $M_1$ to another marking $M_2$, defined by: $\forall p \in P : M_2(p) = (M_1(p) - W(p, tg)) + W(tg, p)$.*

### 2.2 Modular State Spaces

In this section, we will recall the formal definitions of the modular state space from [LP04].

When building the modular state space, we will use Strongly Connected Components. The set of all strongly connected components is denoted by $SCC$. We use $v^c$ to denote the component to which a node $v$ belongs.

We denote the set of states reachable from $M$ by occurrences of local (non-fused) transitions only, in all the individual modules, by $[[M\rangle$.

The notation with a subscript $s$ means the restriction to module $s$, e.g. $[M\rangle_s$ is the set of all nodes reachable from global marking $M$ by occurrences of transitions in module $s$ only.

We use $M_1[[\sigma\rangle\rangle M_2$ to denote that $M_2$ is reachable from $M_1$ by a sequence $\sigma \in (T \setminus TF)^* TF$ of internal transitions followed by a fused transition.

For any reachable marking $M$, we use $M^{\mathscr{A}}$ to denote the product (or tuple) of Strongly Connected Components (SCCs) $M_s^c$ of the individual modules:

$$\forall M \in [M_0\rangle : M^{\mathscr{A}} = \prod_{s \in S} M_s^c.$$

The definition of a modular state space consists of two parts: the state spaces of the individual modules and the synchronisation graph.

**Definition 4 ([LP04], definition 7).** *Let $MN = (S, TF)$ be a modular Petri net with the initial marking $M_0$. The* modular state space *of MN is a pair $MSS = ((SS_s)_{s \in S}, SG)$, where:*

1. *$SS_s = (V_s, A_s)$ is the* local state space *of module $s$:*
   (a) $V_s = \bigcup_{v \in (V_{SG})_s} [v\rangle_s$.
   (b) $A_s = \{(M_1, t, M_2) \in V_s \times (T \setminus TF)_s \times V_s \mid M_1[t\rangle M_2\}$.
2. *$SG = (V_{SG}, A_{SG})$ is the* synchronisation graph *of MN:*
   (a) $V_{SG} = [[M_0\rangle\rangle^{\phi} \cup \{M_0^{\phi}\}$.
   (b) $A_{SG} = \{(M_1^{\phi}, (M_1'^{\phi}, tf), M_2^{\phi}) \in V_{SG} \times ([M_0\rangle^{\phi} \times TF) \times V_{SG} \mid M_1' \in [[M_1\rangle \wedge M_1'[tf\rangle M_2\}$.

*Explanation*
(1) The definition of the state space graphs of the modules is a generalisation of the usual definition of state spaces.

(1a) The set of nodes of the state space graph of a module contains all states locally reachable from any node of the synchronisation graph.

(1b) Likewise, the arcs of the state space graph of a module correspond to all enabled internal transitions of the module.
(2) Each node of the synchronisation graph is labelled by a $M^{\phi}$ and is a representative for all the nodes reachable from $M$ by occurrences of local transitions only, i.e. $[[M\rangle$. The synchronisation graph contains the information on the nodes reachable by occurrences of fused transitions.

(2a) The nodes of the synchronisation graph represent all markings reachable from another marking by a sequence of internal transitions followed by a fused transition. The initial node is also represented.

(2b) The arcs of the synchronisation graph represent all occurrences of fused transitions.

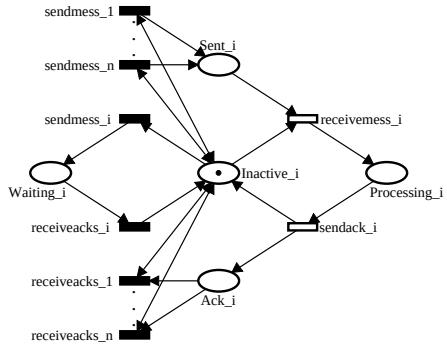
The state space graphs of the modules only contain local information, i.e. the markings of the module and the arcs corresponding to local transitions but not the arcs corresponding to fused transitions. All the information concerning these is stored in the synchronisation graph.

## 3 Case Studies

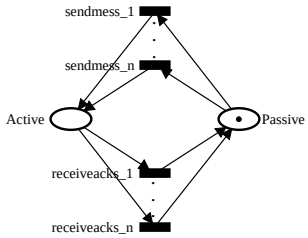
In this section, we describe the case studies that have been used for experimenting the Modular State Space technique.

### 3.1 Distributed Database

The cover picture of this Petrinet newsletter represents a model derived from the distributed database presented in [Jen92] (for 3 database managers on the cover picture). The original coloured Petri net has been unfolded into a Modular Petri net, where a net as described in figure 1(a) is associated with each database manager, while the whole system is synchronised through the net in figure 1(b). The full transitions with the same name are those to be fused, i.e. they form a transition fusion set. They are used for a database manager $i$ to send a message to all other database managers, or to receive all the acknowledgements from these. Note that the transition is different for database manager $i$ from those for database managers $j, \forall j \neq i$.




(a) Database manager $i$




(b) System control

**Fig. 1.** The modular distributed database

### 3.2 Automated Guided Vehicles

The Automated Guided Vehicles (AGVs) problem, in figure 2, was introduced in [KH91]. It has been solved by means of Modular State Spaces in [LP04].

The problem is that of a factory floor which consists of three workstations which operate on parts, two input and one output stations, and five AGVs which move parts from one station to another. The various stations appear on the edges of the net. The two input stations are the subsystems consisting of sets of places {I1, I2} and {I3, I4} (and their neighbouring transitions). The three workstations are captured by the subsystems with sets of places {W11, ..., W14}, {W21, ..., W24} and {W31, ..., W36}. The output station is the subsystem consisting of places {O1, O2}. The subsystems for the various AGVs are modelled by the central parts of the net. Thus vehicle A is captured by the places {A1, ..., A6} and commutes between input station 1 and workstation 1. Four other vehicles (B, D, E and F) travel on the factory floor.

The greyed boxes represent dangerous zones, i.e. areas where the presence of multiple AGVs will lead to a collision. The factory floor, as shown, does not directly exhibit controls of the AGVs. However, it is intended that the filled transitions represent possible control points. In other words, some controller can inhibit the firing of these transitions and thereby prevent collisions from occurring between the AGVs. The other transitions are not controllable, but can provide sensory information about the progress of the AGVs. It is then part of the problem to design the logic of the controller so as to eliminate the possibility of collisions, while minimising the disruption to the system. In other words, it is desirable to retain as much concurrent activity as possible, without allowing collisions to occur.

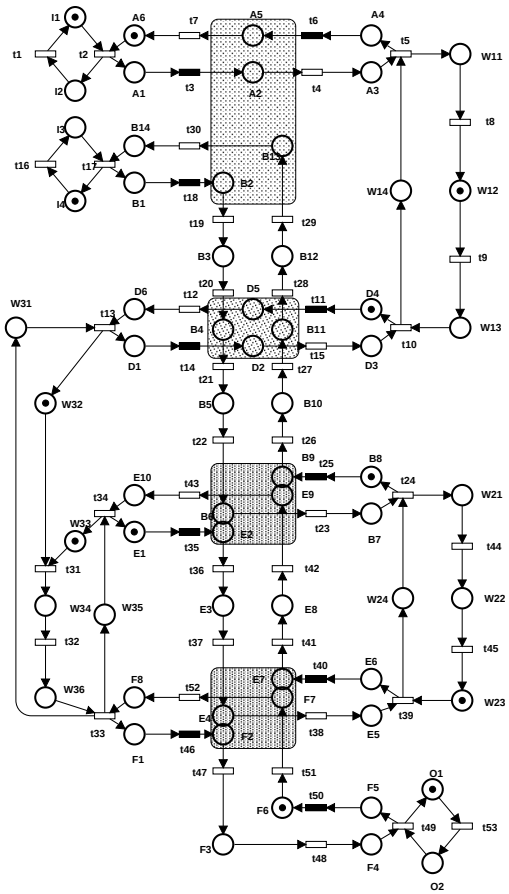
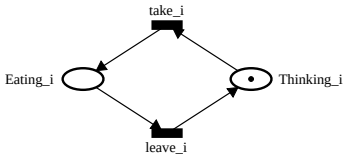


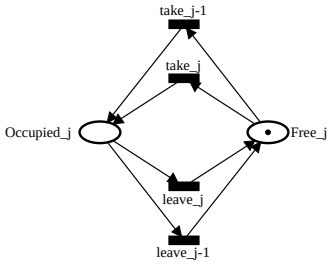

**Fig. 2:** The five AGVs problem.

### 3.3 Philosophers

We considered two versions of the philosophers problem: the usual one, depicted in figure 3, and the poisoned philosophers from [CPN], shown in figure 4: a philosopher can die while eating and then his corpse decomposes.

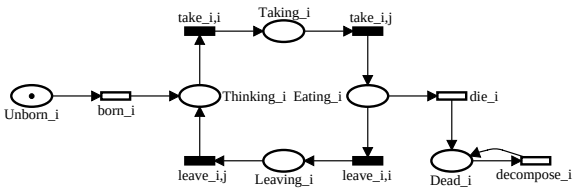



(a) Classical philosopher $i$

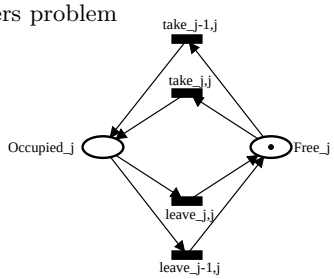



(b) Chopstick $j$

**Fig. 3.** The classical philosophers problem




(a) Poisoned philosopher $i$

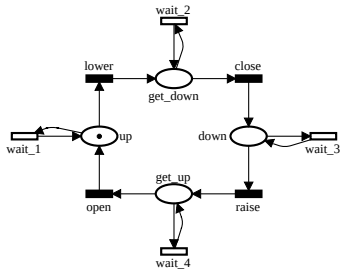



(b) Chopstick $i$

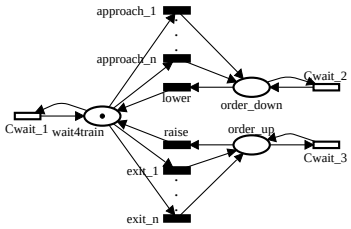
**Fig. 4.** The poisoned philosophers problem

### 3.4 Railway crossing

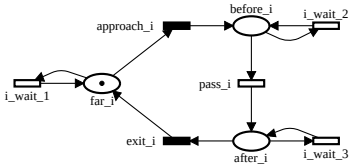
The last example is derived from the classical railway crossing problem. The version considered is presented in figure 5. It is untimed but a transition simulating waiting in a state has been added, and several trains can travel.




(a) Gate




(b) Controller




(c) Train $i$

**Fig. 5.** The railway crossing system

## 4 Experiments

Experiments have been conducted with a prototype tool using the algorithms described in [LP04] in order to obtain the sizes (number of nodes and arcs) of the modular state space. The size of the (flat) occurrence graph is also given, either generated by a tool or calculated. We expressed, when possible the numbers as a function of the parameter $n$ (number of trains, philosophers, ...). Those for the occurrence graph of the database and the philosophers models were already given in [Jen94].

The 5 AGVs example is loosely coupled. Therefore, much interleaving is avoided when building the modular state space and this leads to very good results. On the contrary, the traditional philosophers' example is strongly synchronised. All the transitions then appear in the synchronisation graph which has exactly the same size as the occurrence graph. The local graphs induce additional local nodes. The modified version of the philosophers introduces local behaviour which leads to better results. A similar remark applies to the railway crossing example.

| Model | param | Occurrence Graph | |
|---|---|---|---|
| | | $N_{OG}$ | $A_{OG}$ |
| 5 AGVs | | $30,965,760$ | $345,784,320$ |
| Database | $n$ | $n \times 3^{n-1}$ | $2(n-1) \times 3^{n-2} + 2n$ |
| Philosophers | 2 | 3 | 4 |
| | 3 | 4 | 6 |
| | $n$ | $N_{OG}(n-1) + N_{OG}(n-2)$ | $2n \times F_n, F_n = F_{n-1} + F_{n-2}, F_2 = F_3 = 1$ |
| Poisoned philosophers | 2 | 21 | 38 |
| | 3 | 99 | 264 |
| | $n$ | $4N_{OG}(n-1) + 3N_{OG}(n-2) + 6$ | |
| Railway | $n$ | $4(n^2 + n + 1)$ | $4n^3 + 22n^2 + 16n + 11$ |

**Table 1.** Occurrence graphs

| Model | param | Modular State Space | |
|---|---|---|---|
| | | $N_{MSS}$ | $A_{MSS}$ |
| 5 AGVs | | 900 | 2,687 |
| Database | $n$ | $6n + 3$ | $4n$ |
| Philosophers | 2 | 11 | 4 |
| | 3 | 16 | 6 |
| | $n$ | $N_{OG}(n) + 4n$ | $A_{OG}(n)$ |
| Poisoned philosophers | 2 | 33 | 30 |
| | 3 | 99 | 171 |
| | $n$ | $4N_{MSS}(n-1) + N_{MSS}(n-2) + 8n + 4$ | |
| Railway | $n$ | $\frac{n(n+1)}{2} + 5n + 10$ | $n^2 + 8n + 10$ |

**Table 2.** Modular State Spaces

Finally the cover picture database example gives a significant reduction in the state space size. That might seem surprising at first, as there are very few local transitions w.r.t. synchronised ones. But with a closer look, we notice that the initial marking enables only $n$ synchronised transitions (`sendmess_i`), and then local behaviour takes place, until the corresponding `receiveacks_i` occurs, which leads back to the initial marking. Hence, even though there are many synchronised transitions, all of them quasi-live, they cannot be fired that often.

## References


[CKM01] S. Christensen, L. M. Kristensen, and T. Mailund. A sweep-line method for state space exploration. In *Proc. 7th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2001), Genova, Italy, Apr. 2001*, volume 2031 of *Lecture Notes in Computer Science*, pages 450–464. Springer, 2001.

[CP00] S. Christensen and L. Petrucci. Modular analysis of Petri nets. *The Computer Journal*, 43(3):224–242, 2000.

[CPN] DESIGN/CPN *online*. `http://www.daimi.au.dk/designCPN`.

[Hol91] G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall Int., 1991.

[Jen92] K. Jensen. *Coloured Petri Nets: Basic concepts, analysis methods and practical use. Volume 1: basic concepts*. Monographs in Theoretical Computer Science. Springer, 1992.

[Jen94] K. Jensen. *Coloured Petri Nets: Basic concepts, analysis methods and practical use. Volume 2: analysis methods*. Monographs in Theoretical Computer Science. Springer, 1994.

[KH91] B. Krogh and L. Holloway. Synthesis of feedback control logic for discrete manufacturing systems. *Automatica*, 27(4), 1991.

[LP04] C. Lakos and L. Petrucci. Modular analysis of systems composed of semiautonomous subsystems. In *Proc. 4th Int. Conf. on Application of Concurrency to System Design (ACSD'04), Hamilton, Canada, June 2004*, pages 185–194. IEEE Comp. Soc. Press, June 2004.